A Sequential Importance Sampling Algorithm for Generating Random Graphs with Prescribed Degrees

June 2006

A SEQUENTIAL IMPORTANCE SAMPLING ALGORITHM FOR GENERATING RANDOM GRAPHS WITH PRESCRIBED DEGREES

By Joseph Blitzstein and Persi Diaconis*

Stanford University

Random graphs with a given degree sequence are a useful model capturing several features absent in the classical Erdős-Rényi model, such as dependent edges and non-binomial degrees. In this paper, we use a characterization due to Erdős and Gallai to develop a sequential algorithm for generating a random labeled graph with a given degree sequence. The algorithm is easy to implement and allows surprisingly efficient sequential importance sampling. Applications are given, including simulating a biological network and estimating the number of graphs with a given degree sequence.

1. Introduction. Random graphs with given vertex degrees have recently attracted great interest as a model for many real-world complex networks, including the World Wide Web, peer-to-peer networks, social networks, and biological networks. Newman [58] contains an excellent survey of these networks, with extensive references. A common approach to simulating these systems is to study (empirically or theoretically) the degrees of the vertices in instances of the network, and then to generate a random graph with the appropriate degrees. Graphs with prescribed degrees also appear in random matrix theory and string theory, which can call for large simulations based on random k-regular graphs. Throughout, we are concerned with generating *simple* graphs, i.e., no loops or multiple edges are allowed (the problem becomes considerably easier if loops and multiple edges are allowed).

The main result of this paper is a new sequential importance sampling algorithm for generating random graphs with a given degree sequence. The idea is to build up the graph sequentially, at each stage choosing an edge from a list of candidates with probability proportional to the degrees. Most previously studied algorithms for this problem sometimes either get stuck or produce loops or multiple edges in the output, which is handled by starting over and trying again. Often for such algorithms, the probability of a restart being needed on a trial rapidly approaches 1 as the degree parameters grow, resulting in an enormous number of trials being needed on average to obtain a simple graph. A major advantage of our algorithm is that it never gets stuck. This is achieved using the Erdős-Gallai characterization, which is explained in Section 2, and a carefully chosen order of edge selection.

*Research supported by NSF grants DMS 0072360, 1-24685-1-QABKW, and 1088720-100-QALGE.

AMS 2000 subject classifications: 05C07, 05C80, 68W20, 65C05.

Keywords and phrases: graphical degree sequences, random graphs, random networks, randomized generating algorithms, exponential models, sequential importance sampling.

For example, the graph in Figure 1 is the observed food web of 33 types of organisms (such as bacteria, oysters, and catfish) in the Chesapeake Bay during the summer. The data is from Baird and Ulanowicz [5] and is available online at [77]. Each vertex represents one of the 33 types of organisms, and an edge between two vertices indicates that one preys upon the other. We represent this as an undirected graph for this example, though it is clearly more natural to use a directed graph: it matters whether x eats y or y eats x (especially to x and y). Nevertheless, we will see in Section 11 that the undirected food web reveals interesting information about the connectivity and common substructures in the food web. The blue crab, which is cannibalistic, is represented by vertex 19; we have omitted the loop at vertex 19, not for any moral reason but because we are considering simple graphs.

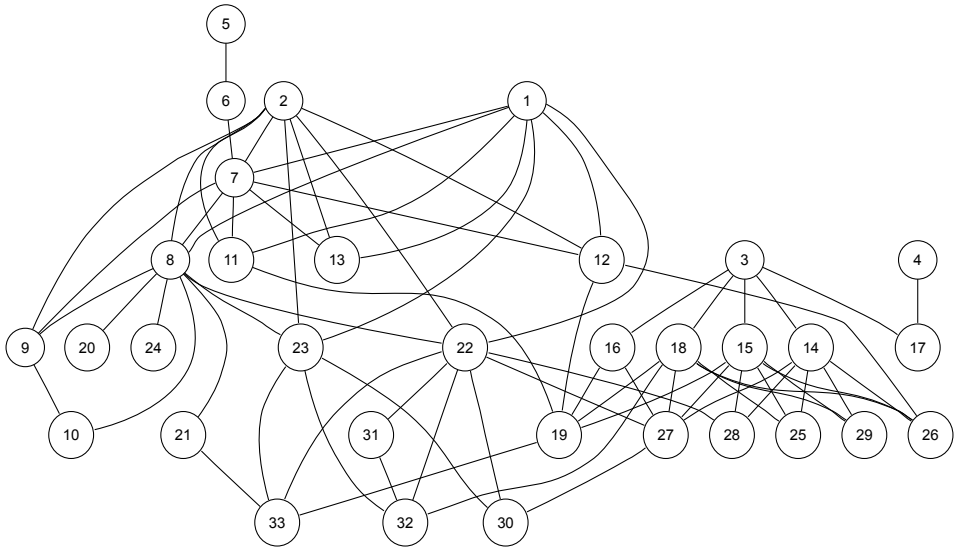

FIG 1. *food web for the Chesapeake Bay ecosystem in summer*

The degree sequence of the above graph is

$$d = (7, 8, 5, 1, 1, 2, 8, 10, 4, 2, 4, 5, 3, 6, 7, 3, 2, 7, 6, 1, 2, 9, 6, 1, 3, 4, 6, 3, 3, 3, 2, 4, 4).$$

Applying importance sampling as explained in Section 9, with 100,000 trials, gave $(1.533 \pm 0.008) \times 10^{57}$ as the estimated number of graphs with the same degree sequence d.

A natural way to test properties of the food web is to condition on the degree sequence, generate a large number of random graphs with the same degree sequence, and then see how the actual food web compares. See Section 11 for details of such a conditional test for this example, using 6000 random graphs with degree sequence d generated by our algorithm.

Section 3 reviews several previous algorithms for our problem. There has been extensive recent development of algorithms for generating random graphs with a given degree

distribution or given *expected* degrees. Britton, Deijfen and Martin-Löf [15] give several algorithms which they show asymptotically produce random graphs with a given degree distribution. Chung and Lu [19, 20] analyze random graphs with given expected degrees (with loops allowed). However, these algorithms and models are not suitable for applications where exactly specified degrees are desired, such as generating or enumerating random k-regular graphs or the model-testing applications of Section 11.

Section 4 presents our algorithm, and Section 5 gives a proof that the algorithm works. The algorithm relies on an extension of the Erdős-Gallai Theorem handling the case where certain edges are *forced* (required to be used); this is discussed in Section 6. This is followed in Section 7 by some estimates of the running time. Section 10 specializes to random trees with a given degree sequence.

The probability of a given output for our algorithm is explicitly computable. The output distribution is generally non-uniform, but the random graphs produced can be used to simulate a general distribution via importance sampling. These ideas are discussed in Section 8, which reviews the literature on importance sampling and sequential importance sampling and shows how they can be used with our algorithm. Applications to approximately enumerating graphs with a given degree sequence are then given in Section 9.

Lastly, in Section 11 we describe an exponential family where the degrees are sufficient statistics, and use the algorithm to test this model on the food web example.

2. Graphical Sequences.

DEFINITION 1. *A finite sequence $(d_1, \ldots, d_n)$ of nonnegative integers ($n \geq 1$) is called* graphical *if there is a labeled simple graph with vertex set $\{1, \ldots, n\}$, in which vertex i has degree d_i. Such a graph is called a* realization *of the degree sequence $(d_1, \ldots, d_n)$. We will call the sequence $(d_1, \ldots, d_n)$ a* degree sequence *regardless of whether or not it is graphical.*

Graphical sequences are sometimes also called *graphic* or *realizable*. Note that if $(d_1, \ldots, d_n)$ is graphical, then $\sum_{i=1}^n d_i$ is even since in any graph, the sum of the degrees is twice the number of edges. Also, it is obviously necessary that $0 \leq d_i \leq n-1$ for all i; the extremes $d_i = 0$ and $d_i = n-1$ correspond to an isolated vertex and a vertex connected by edges to all other vertices, respectively.

There are many well-known efficient tests of whether a sequence is graphical. For example, Mahadev and Peled [51] list eight equivalent necessary and sufficient conditions for graphicality. The most famous criterion for graphicality is due to Erdős and Gallai [27]:

THEOREM 1 (Erdős-Gallai). *Let $d_1 \geq d_2 \geq \cdots \geq d_n$ be nonnegative integers with $\sum_{i=1}^n d_i$ even. Then $d = (d_1, \ldots, d_n)$ is graphical if and only if*

$$\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(k, d_i) \text{ for each } k \in \{1, \ldots, n\}.$$

The necessity of these conditions is not hard to see: for any set S of k vertices in a realization of d, there are at most $\binom{k}{2}$ "internal" edges within S, and for each vertex $v \notin S$, there are at most $\min(k, \deg(v))$ edges from v into S. The sufficiency of the Erdős-Gallai conditions is more difficult to show. Many proofs have been given, including the original in Hungarian [27], a proof using network flows in Berge [10], and a straightforward but tedious induction on the sum of the degrees by Choudum [18]. Note that the Erdős-Gallai conditions can be checked using $\Theta(n)$ arithmetic operations and comparisons and $\Theta(n)$ space, as we can compute and cache the n partial sums of the degrees and for each k the largest i with $\min(k, d_i) = k$ (if any exists), and there are n inequalities to check. The Erdős-Gallai conditions have been refined somewhat, e.g., it is easy to show that it only necessary to check up to $k = n-1$ (for $n \geq 2$), and Tripathi and Vijay [76] showed that the number of inequalities to be checked can be reduced to the number of distinct entries in d. However, these reductions still require checking $\Theta(n)$ inequalities in general. Also note that d needs to be sorted initially if not already given in that form. This can be done in $O(n \log n)$ time using a standard sorting algorithm such as merge-sort.

Instead of having to test all of the inequalities in the Erdős-Gallai conditions, it is often convenient to have a recursive test. A particularly simple recursive test was found independently by Havel [36] and Hakimi [34]. We include the proof since it is instructive and will be useful below.

THEOREM 2 (Havel-Hakimi). *Let d be a degree sequence of length $n \geq 2$ and let i be a coordinate with $d_i > 0$. If d does not have at least d_i positive entries other than i, then d is not graphical. Assume that there are at least d_i positive entries other than at i. Let $\tilde{d}$ be the degree sequence of length $n-1$ obtained from d by deleting coordinate i and subtracting 1 from the d_i coordinates in d of highest degree, aside from i itself. Then d is graphical if and only if $\tilde{d}$ is graphical. Moreover, if d is graphical, then it has a realization in which vertex i is joined to any choice of the d_i highest degree vertices other than vertex i.*

PROOF. If d does not have at least d_i positive entries other than i, then there are not enough vertices to attach i to, so d is not graphical. So assume that there are at least d_i positive entries aside from i. It is immediate that if $\tilde{d}$ is graphical, then d is graphical: take a realization of $\tilde{d}$ (with labels $(1, 2, \ldots, i-1, i+1, \ldots, n)$), introduce a new vertex labeled i, and join i to the d_i vertices whose degrees had 1 subtracted from them.

Conversely, assume that d is graphical, and let G be a realization of d. Let $h_1, \ldots, h_{d_i}$ be a choice of the d_i highest degree vertices other than vertex i (so $\deg(h_j) \geq \deg(v)$ for all $v \notin \{i, h_1, \ldots, h_{d_i}\}$). We are done if vertex i is already joined to all of the h_j, since then we can delete vertex i and its edges to obtain a realization of $\tilde{d}$. So assume that there are vertices v and w (not equal to i) such that $w = h_j$ for some j, v is not equal to any of $h_1, \ldots, h_{d_i}$, and i is adjacent to v but not to w.

If $\deg(v) = \deg(w)$, we can interchange vertices v and w without affecting any degrees. So assume that $\deg(v) < \deg(w)$. Then there is a vertex $x \neq v$ joined by an edge to w

but not to v. Perform a *switching* by adding the edges $\{i, w\}, \{x, v\}$ and deleting the edges $\{i, v\}, \{w, x\}$. This does not affect the degrees, so we still have a realization of d. Repeating this if necessary, we can obtain a realization of d where vertex i is joined to the d_i highest degree vertices other than i itself. □

The Havel-Hakimi Theorem thus gives a recursive test for whether d is graphical: apply the theorem repeatedly until either the theorem reports that the sequence is not graphical (if there are not enough vertices available to connect to some vertex) or the sequence becomes the zero vector (in which case d is graphical). In practice, this recursive test runs very quickly since there are at most n iterations, each consisting of setting a component i to 0 and subtracting 1 from d_i components. The algorithm also needs to find the highest degrees at each stage, which can be done by initially sorting d (in time $O(n \log n)$) and then maintaining the list in sorted order.

Note that when d is graphical, the recursive application of Havel-Hakimi constructs a realization of d, by adding at each stage the edges corresponding to the change in the d vector. This is a simple algorithm for generating a deterministic realization of d. In the next section, we survey previous algorithms for obtaining random realizations of d.

3. Previous Algorithms.

3.1. *Algorithms for random graphs with given degree distributions.* In the classical random graph model $G(n, p)$ of Erdős and Rényi, generating a random graph is easy: for each pair of vertices i, j, independently flip a coin with probability of heads p, and put an edge $\{i, j\}$ iff heads is the result. Thus, the degree of a given vertex has a Binomial$(n - 1, p)$ distribution. Also, note that all vertices have the same expected degree. Thus, a different model is desirable for handling dependent edges and degree distributions that are far from binomial.

For example, many researchers have observed that the Web and several other large networks obey a *power-law degree distribution* (usually with an exponential cutoff or truncation at some point), where the probability of a vertex having degree k is proportional to $k^{-\alpha}$ for some positive constant α. See [2], [6], [14], [15], [22], and [58] for more information about power-law graphs, where they arise, and ways of generating them.

Our algorithm can also be used to generate power-law graphs or, more generally, graphs with a given degree distribution. To do this, first sample i.i.d. random variables $(D_1, \ldots, D_n)$ according to the distribution. If $(D_1, \ldots, D_n)$ is graphical, use it as input to our algorithm; otherwise, re-pick $(D_1, \ldots, D_n)$. Recent work of Arratia and Liggett [4] gives the asymptotic probability that $(D_1, \ldots, D_n)$ is graphical. In particular, the asymptotic probability is $1/2$ if the distribution has finite mean and is not supported on only even degrees or on only odd degrees; clearly, these conditions hold for power-law distributions.

3.2. *The Pairing Model.* Returning to a fixed degree sequence, several algorithms for generating a random (uniform or near-uniform) graph with desired degrees have been

studied. Most of the existing literature has concentrated on the important case of *regular* graphs (graphs where every vertex has the same degree), but some of the results extend to general degree sequences. In the remainder of this section, we will briefly survey these existing algorithms; more details can be found in the references, including the excellent survey on random regular graphs by Wormald [83].

The first such algorithm was the *pairing model* (also known, less descriptively, as the *configuration model*). This model is so natural that it has been re-discovered many times in various forms (see [83] for discussion of the history of the pairing model), but in this context the first appearances seem to be in Bollobás [13] and in Bender and Canfield [9]. Fix n (the number of vertices) and d (the degree) with nd even, and let $v_1, \dots, v_n$ be disjoint *cells*, each of which consists of d points (for general degree sequences, we can let v_i consist of d_i points). Choose a perfect matching of these nd points uniformly. This can be done easily, e.g., by at each stage randomly picking two unmatched points and then matching them together. The matching induces a multigraph, by viewing the cells at vertices and putting an edge between cells v and w for each occurrence of a match between a point in v and a point in w. Under the convention that each loop contributes 2 to the degree, this multigraph is d-regular. The pairing model algorithm is then to generate random multigraphs in this way until a simple graph G is obtained.

Note that the resulting G is uniformly distributed since each simple graph is induced by the same number of perfect matchings. Some properties of Erdős-Rényi random graphs have analogues in this setting. For example, Molloy and Reed [56] use the pairing model to prove the emergence of a giant component in a random graph with a given *asymptotic* degree sequence, under certain conditions on the degrees.

Clearly, the probability $P(\text{simple})$ of a trial resulting in a simple graph is critical for the practicality of this algorithm: the expected number of matchings that need to be generated is $1/P(\text{simple})$. Unfortunately, as d increases, the probability of having loops or multiple edges approaches 1 very rapidly. In fact, Bender and Canfield showed that for fixed d,

$$P(\text{simple}) \sim e^{\frac{1-d^2}{4}} \text{ as } n \to \infty.$$

For very small d such as $d = 3$, this is not a problem and the algorithm works well. But for larger d, the number of repetitions needed to obtain a simple graph becomes prohibitive. For example, for $d = 8$, about 6.9 million trials are needed on average in order to produce a simple graph by this method (for n large).

3.3. *Algorithms Based on the Pairing Model.* A sensible approach is to modify the pairing model by forbidding any choices of matching that will result in a loop or multiple edges. The simplest such method is to start with an empty graph and randomly add edges one at a time, choosing which edge to add by picking uniformly a pair of vertices which have not yet received their full allotment of edges. The process stops when no more edges can be added. With d the maximum allowable degree of each vertex, this is known as a

random d-process. Erdős asked about the asymptotic behavior of the d-process as $n \to \infty$ with d fixed. Ruciński and Wormald [65] answered this by showing that with probability tending to 1, the resulting graph is d-regular (for nd even).

A similar result has been obtained by Robalewska and Wormald [63] for the *star d-process*, which at each stage chooses a vertex with a minimal current number δ of edges, and attempts to connect it to $d - \delta$ allowable vertices. However, for both the d-process and the star d-process, it is difficult to compute the resulting probability distribution on d-regular graphs. Also, although it is possible to consider analogous processes for a general degree sequence $(d_1, \ldots, d_n)$, little seems to be known in general about the probability of the process succeeding in producing a realization of $(d_1, \ldots, d_n)$.

A closely related algorithm for d-regular graphs is given by Steger and Wormald [71], designed to have an approximately uniform output (for large n, under certain growth restrictions on d). Their algorithm proceeds as in the pairing model, except that before deciding to make a match with two points i and j, one first checks that they are not in the same cell and that there is not already a match between a cellmate of i and a cellmate of j. The algorithm continues until no more permissible pairs can be found. This can be viewed as a variant of the d-process with edges chosen with probabilities depending on the degrees, rather than uniformly.

By construction, the Steger-Wormald algorithm avoids loops and multiple edges, but unlike the pairing model it can get stuck before a perfect matching is reached, as there may be unmatched vertices left over which are not allowed to be paired with each other. If the algorithm gets stuck in this way, it simply starts over and tries again. Steger and Wormald showed that the probability of their algorithm getting stuck approaches 0 for $d = o((n/(\log^3 n)^{1/11})$, and Kim and Vu [42] recently improved this bound to $d = o(n^{1/3}/\log^2 n)$. The average running time is then $O(nd^2)$ for this case. Empirically, Steger and Wormald observed that their algorithm seems to get stuck with probability at most 0.7 for $d \le n/2$, but this has not been proved for d a sizable fraction of n. The output of the Steger-Wormald algorithm is not uniform, but their work and later improvements by Kim and Vu [41] show that for $d = o(n^{1/3-\epsilon})$ with $\epsilon > 0$ fixed, the output is asymptotically uniform. For d of higher order than $n^{1/3}$, it is not known whether the asymptotic distribution is close to uniform. Again these results are for regular graphs, and it is not clear how far they extend to general degree sequences.

There is also an interesting earlier algorithm by McKay and Wormald [53] based on the pairing model. Their algorithm starts with a random pairing from the pairing model, and then uses two types of switchings (slightly more complicated than the switchings in the Markov chain described below). One type of switching is used repeatedly to eliminate any loops from the pairing, and then the other type is used to eliminate any multiple edges from the pairing. To obtain an output graph which is uniformly distributed, an accept/reject procedure is used: in each iteration, a restart of the algorithm is performed with a certain probability. Let $M = \sum_{i=1}^n d_i$, $M_2 = \sum_{i=1}^n d_i(d_i - 1)$, and $d_{\max} = \max\{d_1, \ldots, d_n\}$. McKay and Wormald show that if $d_{\max}^3 = O(M^2/M_2)$ and $d_{\max}^3 = o(M + M_2)$, then the average

running time of their algorithm is $O(M + M_2^2)$. Note that for d-regular graphs, this reduces to an average running time of $O(n^2 d^4)$ for $d = O(n^{1/3})$. They also give a version that runs in $O(nd^3)$ time under the same conditions, but Wormald says in [83] that implementing this version is "the programmer's nightmare."

3.4. *Adjacency Lists and Havel-Hakimi Variants.* Tinhofer [74, 75] gave a general algorithm for generating random graphs with given properties, including given degree sequences. This approach involves choosing random *adjacency lists*, each of which consists of vertices adjacent to a particular vertex. These are chosen so that each edge is represented in only one adjacency list, to avoid redundancy. An accept/reject procedure can then be used to obtain a uniform output. However, this algorithm is quite complicated to analyze, and neither the average running time nor the correct accept/reject probabilities seem to be known.

As noted earlier, the Havel-Hakimi Theorem gives a deterministic algorithm for generating a realization of d. A simple modification to add randomness is as follows. Start with vertices $1, \ldots, n$ and no edges. At each stage, pick a vertex i with $d_i > 0$ and choose d_i other vertices to join i to, according to some pre-specified probabilities depending on the degrees. Obtain d' from d by setting the entry at position i to 0 and subtracting 1 from each chosen vertex. If d' is graphical, update d to d' and continue; otherwise, choose a different set of vertices to join i to. If favoritism is shown towards higher degree vertices, e.g., by choosing to connect i to a legal j with probability proportional to the current degree of j, then this algorithm becomes similar to the preferential attachment model discussed in Barabási and Albert [6]. However, we do not have good bounds on the probability of a chosen set of vertices being allowable, which makes it difficult to analyze the average running time. Also, to use this algorithm with the importance sampling techniques given later, it seems necessary to know at each stage exactly how many choices of the d_i vertices yield a graphical d'; but it is obviously very undesirable to have to test all subsets of a certain size.

3.5. *Markov Chain Monte Carlo Algorithms.* A natural, widely-used approach is to use a Markov chain Monte Carlo (MCMC) algorithm based on the *switching* moves used in the proof of the Havel-Hakimi Theorem. Let $d = (d_1, \ldots, d_n)$ be graphical. We can run a Markov Chain with state space the set of all realizations of d as follows. Start the chain at any realization of d (this can be constructed efficiently using Havel-Hakimi). When at a realization G, pick two random edges $\{x, y\}$ and $\{u, v\}$ uniformly with x, y, u, v distinct. If $\{x, u\}$ and $\{y, v\}$ are not edges, then let the chain go to the realization G' obtained by adding the edges $\{x, u\}, \{y, v\}$ and deleting the edges $\{x, y\}, \{u, v\}$; otherwise, stay at G. (Alternatively, we can also check whether $\{x, v\}$ and $\{y, u\}$ are non-edges; if so and the other switching is not allowed, then use this switching. If both of these switchings are possible, we can give probability 1/2 to each.)

The Markov chain using switchings is irreducible since the proof of Havel-Hakimi can be

used to obtain a sequence of switchings to take any realization of d to any other realization of d. It is also easy to see that the chain is reversible with the uniform distribution as its stationary distribution.

Note that in the context of tables, the corresponding random walk on adjacency matrices uses exactly the same moves as the well-known random walk on contingency tables or zero-one tables discussed in Diaconis and Gangolli [23], where at each stage a pair of random rows and a pair of random columns are chosen, and the current table is modified in the four entries according to the pattern $\begin{pmatrix} + & - \\ - & + \end{pmatrix}$ or the pattern $\begin{pmatrix} - & + \\ + & - \end{pmatrix}$, if possible. Diaconis and Sturmfels [24] have generalized these moves to discrete exponential families, which provides further evidence of the naturalness of this chain.

For regular graphs, the mixing time of the switchings Markov chain has been studied very recently by Cooper, Dyer, and Greenhill [21], using a path argument. They also use the switchings chain to model a peer-to-peer network, which is a decentralized network where users can share information and resources. The vertices of the graph correspond to the users, and two users who are joined by an edge can exchange information. Using random regular graphs is a flexible, convenient way to create such a network.

Practitioners sometimes want to generate only *connected* graphs with the given degree sequence. Note that a switching move may disconnect a connected graph. However, if we accept a switching only if the resulting graph is connected, it follows from a theorem of Taylor [73] that the switchings Markov chain on connected graphs is irreducible. To avoid the inefficiency of having to check after every step whether the graph is still connected, Gkantsidis, Mihail, and Zegura [31] and Viger and Latapy [79] propose heuristic modifications and give empirical performance data.

For regular graphs, another MCMC algorithm was given by Jerrum and Sinclair [39]. In their algorithm, a graph is constructed in which perfect matchings give rise to simple graphs. The algorithm then uses a Markov chain to obtain a perfect matching in this graph. Jerrum and Sinclair showed that this algorithm comes arbitrarily close to uniform in polynomial time (in n). However, the polynomial is of fairly high order. Their algorithm can be extended to non-regular graphs, but is only known to run in polynomial time if d satisfies a condition called *P-stability*. Intuitively, a class of degree sequences is P-stable if a small perturbation of a degree sequence d in the class does not drastically change the number of graphs with that degree sequence. For precise details and conditions for P-stability in terms of the minimum and maximum degrees, see Jerrum, Sinclair and McKay [40].

4. The Sequential Algorithm. In this section, we present our sequential algorithm and describe some of its features. In a graph process aiming to produce a realization of a given degree sequence, the word "degree" could either mean the number of edges chosen already for some vertex, or it could mean the *residual degree*, which is the remaining number of edges which must be chosen for that vertex. In discussing our algorithm, degrees should be thought of as residual degrees unless otherwise specified; in particular, we will start with

the degree sequence d and add edges until the degree sequence is reduced to 0.

As we will be frequently adding or subtracting 1 at certain coordinates of degree sequences, we introduce some notation for this operation.

NOTATION 1. *For any vector $d = (d_1, \dots d_n)$ and distinct $i_1, \dots, i_k \in \{1, \dots, n\}$, define $\oplus_{i_1,\dots,i_k} d$ to be the vector obtained from d by adding 1 at each of the coordinates $i_1, \dots, i_k$, and define $\ominus_{i_1,\dots,i_k}$ analogously:*

$$(\oplus_{i_1,\dots,i_k} d)_i = \begin{cases} d_i + 1 & \text{for } i \in \{i_1, \dots, i_k\}, \\ d_i & \text{otherwise,} \end{cases},$$

$$(\ominus_{i_1,\dots,i_k} d)_i = \begin{cases} d_i - 1 & \text{for } i \in \{i_1, \dots, i_k\}, \\ d_i & \text{otherwise.} \end{cases}$$

For example, $\oplus_{1,2}(3,2,2,1) = (4,3,2,1)$ and $\ominus_{1,2}(3,2,2,1) = (2,1,2,1)$. With this notation, our sequential algorithm can be stated compactly.

SEQUENTIAL ALGORITHM FOR RANDOM GRAPH WITH GIVEN DEGREES

Input: a graphical degree sequence $(d_1, ..., d_n)$.

1. Let E be an empty list of edges.
2. If $d = 0$, terminate with output E.
3. Choose the least i with d_i a minimal positive entry.
4. Compute candidate list $J = \{j \neq i : \{i, j\} \notin E \text{ and } \ominus_{i,j} d \text{ is graphical}\}$.
5. Pick $j \in J$ with probability proportional to its degree in d.
6. Add the edge $\{i, j\}$ to E and update d to $\ominus_{i,j} d$.
7. Repeat steps 4-6 until the degree of i is 0.
8. Return to step 2.

Output: E.

For example, suppose that the starting sequence is $(3,2,2,2,1)$. The algorithm starts by choosing which vertex to join vertex 5 to, using the candidate list $\{1,2,3,4\}$. Say it chooses 2. The new degree sequence is $(3,1,2,2,0)$. The degree of vertex 5 is now 0, so the algorithm continues with vertex 2, etc. One possible sequence of degree sequences is

$$\begin{aligned}(3,2,2,2,1) \quad &\to \quad (3,1,2,2,0) \to (2,0,2,2,0) \to (1,0,2,1,0) \to (0,0,1,1,0) \\ \to (0,0,0,0,0),&\end{aligned}$$

corresponding to the graph with edge set

$$\{\{5,2\},\{2,1\},\{1,4\},\{1,3\},\{3,4\}\}.$$

As another example, we show below the first two of 6000 graphs generated using our algorithm applied to the degree sequence of the food web of Figure 1. Each took about 13 seconds to generate (on a 1.33 GHz PowerBook). Qualitatively, they appear to be more spread out and less neatly hierarchical than the actual food web. We discuss this more in Section 11, comparing some test statistics of the actual graph with those of the random graphs.

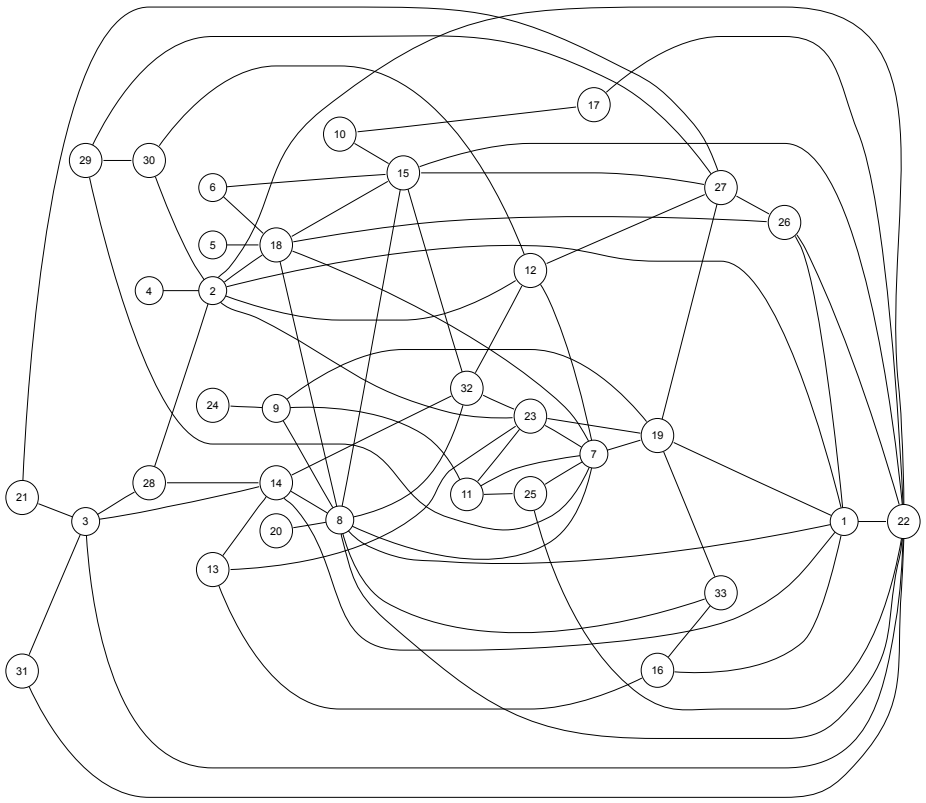

FIG 2. *random graph with food web degrees, 1*

The algorithm always terminates in a realization of $(d_1, \ldots, d_n)$. The output of the algorithm is not uniformly distributed over all realizations of $(d_1, \ldots, d_n)$ in general, but every realization of $(d_1, \ldots, d_n)$ has positive probability. Importance sampling techniques can then be used to compute expected values with respect to the uniform distribution if desired, as described in Section 8.

We now make remarks on the specific steps and some ways of speeding up the implementation of the algorithm.

1. In Step 4, any test for graphicality can be used; Erdős-Gallai is particularly easy to

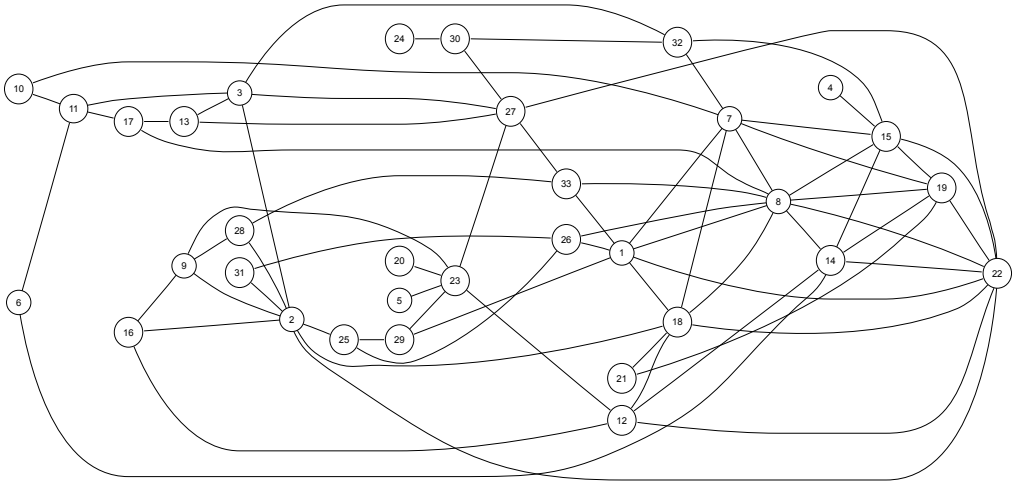

FIG 3. *random graph with food web degrees, 2*

implement and runs quickly.

2. It follows from Theorem 3 that a candidate at a later stage is also a candidate at an earlier stage, within the same choice of i from Step 3. Thus, in Step 4 it is sufficient to test the vertices which were candidates in the previous stage, if that stage had the same choice of i.

3. Let
$$m = |\{j : d_j \geq j - 1\}|$$
be the *corrected Durfee number* of d. A beautiful result (see Theorem 3.4.1 in Mahadev and Peled [51]) is that d is graphical if and only if it satisfies the first m Erdős-Gallai inequalities. In many cases the corrected Durfee number m is much less than n, which greatly reduces the number of inequalities to check.
4. In Step 5, any probability distribution p on J with $p(j) > 0$ for all j can be used. An interesting problem here is to find a distribution p which makes the output as close to uniform as possible. In our empirical tests, choosing a candidate with probability proportional to its degree was significantly better than choosing a candidate uniformly (see Section 8). But it remains open to prove some sort of optimality here. In the case of the algorithm for random trees (see Section 10), picking j with probability proportional to $d_j - 1$ gives an exactly uniform tree.

5. An alternative to Step 4 would be to pick j with $\{i, j\} \notin E$ randomly and accept it if $\ominus_{i,j} d$ is graphical; otherwise, pick again without replacement. This approach

runs faster, but has the disadvantage that it becomes very difficult to compute the sequential probabilities discussed in Section 8.

6. Another alternative would be to connect at each step a highest degree vertex to a randomly chosen candidate. This has the advantage that it is obvious from Havel-Hakimi that the algorithm never gets stuck. However, this seems to make it very difficult to compute the weights $c(Y)$ needed in Section 8 for importance sampling. Choosing the maximum vertex on each step involves a lot of jumping around from vertex to vertex, whereas our main algorithm is more systematic in the sense that it chooses a vertex to fully connect, then chooses another vertex to fully connect, etc.

5. Proof that the Algorithm Works. The theorem below guarantees that the algorithm never gets stuck, by showing that at least one candidate vertex exists at each stage.

THEOREM 3. *Let $d = (d_1, \ldots, d_n)$ be a graphical degree sequence with $d_i > 0$, arranged so that $d_n = \min\{d_1, \ldots, d_n\}$. Let $d = d^{(0)}, d^{(1)}, d^{(2)}, \ldots, d^{(j)} = \tilde{d}$ be graphical degree sequences of length n (for some $j \geq 1$), such that $d^{(i)}$ is obtained from $d^{(i-1)}$ by subtracting 1 at coordinate n and at another coordinate v_i not previously changed. That is,*

$$d^{(i)} = \ominus_{n,v_i} d^{(i-1)} \text{ for } i \in \{1, \ldots, j\},$$

where $n, v_1, \ldots, v_j$ are distinct. Then d has a realization containing all of the edges $\{n, v_1\}, \ldots, \{n, v_j\}$, and $\tilde{d}$ has a realization containing none of these edges.

PROOF. The desired realization of d immediately yields that of $\tilde{d}$, by deleting the edges $\{n, v_1\}, \ldots, \{n, v_j\}$, and conversely the desired realization of $\tilde{d}$ immediately gives that of d. Note that $j \leq d_n$ since the degree of vertex n in $d^{(j)}$ is $d_n - j$. We use a backwards induction on j, over $1 \leq j \leq d_n$, by showing that the claim is true for $j = d_n$ and that if it is true for $j + 1$, then it is true for j.

First assume $j = d_n$, and let G_j be a realization of the degree sequence $d^{(j)}$. Note that vertex n has degree 0 in G_j. Adding edges in G_j from vertex n to each $v_i, 1 \leq i \leq j$, yields a graph with degree sequence $d^{(0)}$ containing the desired edges. Now assume that the result holds for $j + 1$, and show it for j, for some fixed j with $1 \leq j \leq d_n - 1$.

Call the vertices $v_1, \ldots, v_j$ *touched* vertices, and the remaining vertices other than vertex n *untouched*. The proof hinges on whether we can find a realization of $d^{(j)}$ where vertex n is adjacent to an untouched vertex.

Suppose that $d^{(j)}$ has a realization G_j containing an edge $\{n, x\}$, with x an untouched vertex. Deleting the edge $\{n, x\}$ yields a graph G_{j+1} with degree sequence $d^{(j+1)}$ of the form in the statement of the theorem, with $j + 1$ in place of j and $v_{j+1} = x$. The inductive hypothesis then implies that $d^{(0)}$ has a realization containing the edges $\{n, v_1\}, \ldots, \{n, v_{j+1}\}$.

So it suffices to show that $d^{(j)}$ has a realization with an edge $\{n, x\}$, with x an untouched vertex.

Let T consist of the j touched vertices, and decompose $T = A \cup B$, where a touched vertex x is in A iff $\{n, x\}$ is an edge in some realization of $d^{(j)}$, and $B = T \setminus A$. Here B may be empty, but we can assume A is nonempty, since otherwise any realization of $d^{(j)}$ has an edge $\{n, x\}$ with x untouched. Let $|A| = a$ and $|B| = b$ (so $a + b = j$).

Let $x \in A$ and y be an untouched vertex or $y \in B$. Consider a realization of $d^{(j)}$ containing the edge $\{n, x\}$. Note that if the degrees of x and y are equal in $d^{(j)}$, then they can be interchanged without affecting any degrees, and then $\{n, y\}$ is an edge (which contradicts the definition of B if $y \in B$, and gives us the desired edge if y is untouched). If the degree of x is less than that of y, then we can perform a switching as in the proof of the Havel-Hakimi Theorem (pick a vertex $w \neq x$ adjacent to y but not adjacent to x, add edges $\{n, y\}, \{x, w\}$, and delete edges $\{n, x\}, \{y, w\}$), again producing a realization with the edge $\{n, y\}$. So assume that the degree of x is strictly greater than the degree of y in $d^{(j)}$ for all $x \in A$ and y which is either untouched or in B. Then the vertices in A have the a highest degrees (excluding n itself) in $d^{(j)}$.

Let d' be the degree sequence with $d'_n = d_n - b$, $d'_y = d_y - 1$ for $y \in B$, and $d'_y = d_y$ otherwise. Note that $\tilde{d}_i \leq d'_i \leq d_i$ for all i, with equality on the left for $i \in B$ and equality on the right for $i \in A$. Also, the assumption $d_n = \min\{d_1, \ldots, d_n\}$ implies that $d'_n = \min\{d'_1, \ldots, d'_n\}$ and $d_n^{(j)} = \min\{d_1^{(j)}, \ldots, d_n^{(j)}\}$. We claim that d' is graphical. Assuming that d' is graphical, we can then complete the proof as follows. Note that the vertices in A have the a highest degrees in d' (excluding n itself) since this is true for $d^{(j)}$ and in passing from $d^{(j)}$ to d', these degrees are increased by 1 while all other degrees aside from that of vertex n are unchanged. So by the Havel-Hakimi Theorem, d' has a realization containing all of the edges $\{n, x\}, x \in A$ (as $a \leq d'_n = d_n - b$, since $j < d_n$). Deleting these edges yields a realization $G^{(j)}$ of $d^{(j)}$ containing none of these edges. By definition of B, $G^{(j)}$ also does not contain any edge $\{n, y\}$ with $y \in B$. Thus, $G^{(j)}$ is as desired. So it suffices to prove that d' is graphical.

To show that d' is graphical, we check the Erdős-Gallai conditions. For $k = n$, since d is graphical we have

$$\sum_{i=1}^{n} d'_i \leq \sum_{i=1}^{n} d_i \leq n(n-1).$$

Assume $k < n$, and let $I \subseteq \{1, \ldots, n-1\}$ be an index set for the k largest degrees of d'. If

$k \le d'_n$, then $k \le d'_i \le d_i$ for all i, and we have

$$\begin{aligned}\sum_{i\in I} d'_i &\le \sum_{i\in I} d_i \\ &\le k(k-1) + \sum_{i\notin I} \min\{k, d_i\} \\ &= k(k-1) + \sum_{i\notin I} k \\ &= k(k-1) + \sum_{i\notin I} \min\{k, d'_i\}.\end{aligned}$$

So assume $k > d'_n$ (which implies $k > \tilde{d}_n$). Then

$$\sum_{i\in I} d'_i = a' + \sum_{i\in I} \tilde{d}_i,$$

where $a' \le a$, since d' and $\tilde{d}$ differ only on $A \cup \{n\}$. Since $\tilde{d}$ is graphical,

$$\begin{aligned}a' + \sum_{i\in I} \tilde{d}_i &\le a' + k(k-1) + \sum_{i\notin I} \min\{k, \tilde{d}_i\} \\ &= a' + k(k-1) + \tilde{d}_n + \sum_{i\notin I, i\neq n} \min\{k, \tilde{d}_i\} \\ &\le a' + k(k-1) + \tilde{d}_n + \sum_{i\notin I, i\neq n} \min\{k, d'_i\} \\ &= a' + k(k-1) + \tilde{d}_n - d'_n + \sum_{i\notin I} \min\{k, d'_i\} \\ &\le k(k-1) + \sum_{i\notin I} \min\{k, d'_i\}\end{aligned}$$

where the last inequality follows from $a' + \tilde{d}_n - d'_n = a' + (d_n - j) - (d_n - b) = a' - a \le 0$. Hence, d' is graphical.

□

The above result is false if the assumption $d_n = \min\{d_1, \ldots, d_n\}$ is dropped. For a counterexample, consider the degree sequences

$$(1,1,2,2,5,3), (1,1,2,2,4,2), (0,1,2,2,4,1).$$

It is easily checked that they are graphical and each is obtained from the previous one in the desired way. But there is no realization of the sequence $(1,1,2,2,5,3)$ containing the edge $\{1,6\}$, since clearly vertex 1 must be connected to vertex 5. This would result in the algorithm getting stuck in some cases if we did not start with a minimal positive degree

vertex: starting with $(1,1,2,2,5,3)$, the algorithm could choose to form the edge $\{5,6\}$ and then choose $\{1,6\}$, since $(0,1,2,2,4,1)$ is graphical. But then vertex 5 could never achieve degree 4 without creating multiple edges.

Using the above theorem, we can now prove that the algorithm never gets stuck.

COROLLARY 1. *Given a graphical sequence $d = (d_1, \dots, d_n)$ as input, the algorithm above terminates with a realization of d. Every realization of d occurs with positive probability.*

PROOF. We use induction on the number of nonzero entries in the input vector d. If $d = 0$, the algorithm terminates immediately with empty edge set, which is obviously the only realization of the zero vector. Suppose $d \neq 0$ and the claim is true for all input vectors with fewer nonzero entries than d. Let i be the smallest index in d with minimal positive degree. There is at least one candidate vertex j to connect i to, since if $\{i,j\}$ is an edge in a realization of d, then deleting this edge shows that the sequence $d^{(1)}$ obtained by subtracting 1 at coordinates i and j is graphical.

Suppose that the algorithm has chosen edges $\{i,v_1\}, \dots, \{i,v_j\}$ with corresponding degree sequences $d = d^{(0)}, d^{(1)}, \dots, d^{(j)}$, where $j \geq 1$ and $d_i^{(j)} = d_i - j > 0$. Omitting any zeroes in d and permuting each sequence to put vertex i at coordinate n, Theorem 3 implies that $d^{(j)}$ has a realization G_j using none of the edges $\{i,v_1\}, \dots, \{i,v_j\}$. Then G_j has an edge $\{i,x\}$ with $d_x = d_x^{(j)}$, and $\{i,x\}$ is an allowable choice for the next edge. Therefore, the algorithm can always extend the list of degree sequences $d = d^{(0)}, \dots, d^{(j)}$ until the degree at i is $d_i - j = 0$. Let $\{i,v_1\}, \dots \{i,v_j\}$ be the edges selected (with $d_i - j = 0$). Note that if $\{i,w_1\}, \dots \{i,w_j\}$ are the edges incident with i in a realization of G, then these edges are chosen with positive probability (as seen by deleting these edges one by one in any order).

The algorithm then proceeds by picking a minimal positive entry in $d^{(j)}$ (if any remains). By the inductive hypothesis, running the algorithm on input vector $d^{(j)}$ terminates with a realization of $d^{(j)}$. Thus, the algorithm applied to d terminates with edge set $E = \{\{i,v_1\}, \dots \{i,v_j\}\} \cup \tilde{E}$, where $\tilde{E}$ is an output edge set of the algorithm applied to $d^{(j)}$. No edges in $\tilde{E}$ involve vertex i, so E is a realization of d. Again by the inductive hypothesis, every realization of $d^{(j)}$ is chosen with positive probability, and it follows that every realization of d is chosen with positive probability.

□

Although the algorithm always gives a realization of d, for certain degree sequences a prohibitive number of trials would be needed for accurate estimation. On the other hand, for many degree sequences which have arisen in practice, the importance sampling can easily be done efficiently.

As an extreme example, consider d of the form $d = (1, \dots, 1, k)$ with $k \geq 2$, which is the analogue for graphs of an example recently considered by Bezáková, Sinclair, Stefankovic, and Vigoda [11] in the context of the Chen, Diaconis, Holmes, and Liu [16] algorithm for

generating zero-one tables with given row and column sums. Bezáková et al. show that in certain examples with all but one row sum and all but one column sum equal to 1, exponentially many trials are needed for the sequential importance sampling estimate to give a good estimate of the true number of tables. Blitzstein [12] gives explicit variance calculations for $d = (1, \ldots, 1, k)$ with $l + 2k$ 1's, and the relative standard deviation of the importance sampling estimator grows rapidly in k once k exceeds a certain threshold value. For example, when $l = 1$ and $k = 100$, the relative standard deviation is 2.2925×10^{12}, so over 10^{24} graphs would be needed to obtain an acceptably low standard deviation. Further work is required to understand what degree sequences lead to such behavior.

Bayati and Saberi [8] analyze a closely related example for graphs with several $\sqrt{n}$ terms in place of the single k, showing that for the Steger-Wormald algorithm the resulting distribution is extremely far from uniform. They also show how to obtain an improved algorithm by re-weighting the edge selection process.

6. Forced Sets of Edges. Theorem 3 is also related to the problem of finding a realization of a graph which requires or forbids certain edges. To make this precise, we introduce the notion of a forced set of edges.

DEFINITION 2. *Let d be a graphical degree sequence. A set F of pairs $\{i, j\}$ with $i, j \in \{1, \ldots, n\}$ is* forced *for d if for every realization $G = (V, E)$ of d, $F \cap E \neq \emptyset$. If a singleton $\{e_1\}$ is forced for d, we will say that e_1 is a forced edge for d.*

The one-step case ($j = 1$) in Theorem 3 gives a criterion for an edge to be forced. Indeed, for this case the assumption about the minimum is not needed.

PROPOSITION 1. *Let d be a graphical degree sequence and $i, j \in \{1, \ldots, n\}$ with $i \neq j$. Then $\{i, j\}$ is a forced edge for d if and only if $\oplus_{i,j} d$ is not graphical.*

PROOF. Suppose that $\{i, j\}$ is not forced for d. Adding the edge $\{i, j\}$ to a realization of d yields a realization of $\oplus_{i,j} d$. Conversely, suppose that $\{i, j\}$ is forced for d. Arguing as in the proof of Theorem 3, we see that i and j must have greater degrees in d than any other vertex. Suppose (for contradiction) that $\oplus_{i,j} d$ is graphical. Then Havel-Hakimi gives a realization of $\oplus_{i,j} d$ that uses the edge $\{i, j\}$. Deleting this edge gives a realization of d not containing the edge $\{i, j\}$, contradicting it being forced for d. □

Beyond this one-step case, an additional assumption is needed for analogous results on forced sets, as shown by the counterexample after the proof of Theorem 3. Much of the proof consisted of showing that the set of all "touched" vertices is not a forced set for $\tilde{d}$. This immediately yields the following result.

COROLLARY 2. *Let d be a graphical degree sequence and $d' = \oplus_i^k \oplus_{j_1,\ldots,j_k} d$ for some distinct vertices $i, j_1, \ldots, j_k$. Suppose that $d'_i = \min\{d'_1, \ldots, d'_n\}$. Then the set of edges $\{\{i, j_1\}, \ldots, \{i, j_k\}\}$ is forced for d if and only if d' is not graphical.*

We may also want to know whether there is a realization of d containing a certain list of desired edges. This leads to the following notion.

DEFINITION 3. *Let d be a graphical degree sequence. A set S of pairs $\{i,j\}$ with $i,j \in \{1,\dots,n\}$ is* simultaneously allowable *for d if d has a realization G with every element of S an edge in G. If S is a simultaneously allowable singleton, we call it an allowable edge.*

Results on forced sets imply dual results for simultaneously allowable sets and vice versa, by adding or deleting the appropriate edges from a realization. For example, Proposition 1 above implies the following.

COROLLARY 3. *Let d be a graphical degree sequence and $i,j \in \{1,\dots,n\}$ with $i \neq j$. Then $\{i,j\}$ is an allowable edge for d if and only if $\ominus_{i,j}d$ is graphical.*

Similarly, the dual result to Corollary 2 is the following (which is also an easy consequence of Theorem 3).

COROLLARY 4. *Let d be a graphical degree sequence and $\tilde{d} = \ominus_i^k \ominus_{j_1,\dots,j_k} d$ for some distinct vertices $i, j_1, \dots, j_k$. Suppose that $d_i = \min\{d_1,\dots,d_n\}$. Then $\{\{i,j_1\},\dots,\{i,j_k\}\}$ is simultaneously allowable for d if and only if $\tilde{d}$ is graphical.*

7. Running Time. In this section, we examine the running time of the algorithm. Let $d = (d_1,\dots,d_n)$ be the input. Since no restarts are needed, the algorithm has a fixed, bounded worst case running time. Each time a candidate list is generated, the algorithm performs $O(n)$ easy arithmetic operations (adding or subtracting 1) and tests for graphicality $O(n)$ times. Each test for graphicality can be done in $O(n)$ time by Erdős-Gallai, giving a total worst case $O(n^2)$ running time each time a candidate list is generated (a sorted degree sequence also needs to be maintained, but the total time needed for this is dominated by the $O(n^2)$ time we already have).

Since a candidate list is generated for each time an edge is selected, there are $\frac{1}{2}\sum_{i=1}^n d_i$ candidate lists to generate. Additionally, the algorithm sometimes needs to locate the smallest index of a minimal nonzero entry, but we are already assuming that we are maintaining a sorted degree sequence.

The overall worst case running time is then $O(n^2\sum_{i=1}^n d_i)$. For d-regular graphs, this becomes a worst case of $O(n^3 d)$ time. Note that an algorithm which requires restarts has an unbounded worst case running time.

Using Remark 2 after Theorem 3, the number of times that the Erdős-Gallai conditions must be applied is often considerably less than n. But we do not have a good bound on the average number of times that Erdős-Gallai must be invoked, so we do not have a better bound on the average running time than the worst case running time of $O(n^2\sum_{i=1}^n d_i)$.

8. Importance Sampling. The random graphs generated by the sequential algorithms are not generally distributed uniformly, but the algorithm makes it easy to use importance sampling to estimate expected values and probabilities for any desired distribution, including uniform. Importance sampling allows us to re-weight samples from a distribution available to us, called the *trial distribution*, to obtain estimates with respect to the desired *target distribution*. Often choosing a good trial distribution is a difficult problem; one approach to this, sequential importance sampling, involves building up the trial distribution recursively as a product, with each factor conditioned on the previous choices.

Section 8.1 reviews some previous applications of importance sampling and explains how they are related to the current work. Section 8.2 then shows how sequential importance sampling works in the context of random graphs produced by our algorithm.

For general background on Monte Carlo computations and importance sampling, we recommend Hammersley and Handscomb [35], Liu [47], and Fishman [28]. Sequential importance sampling is developed in Liu and Chen [48] and Doucet, de Freitas and Gordon [25]. Important variations which could be coupled with the present algorithms include Owen's [60] use of control variates to bound the variance and adaptive importance sampling as in Rubinstein [64]. Obtaining adequate bounds on the variance for importance sampling algorithms is an open research problem in most cases of interest (see Bassetti and Diaconis [7] for some recent work in this area).

8.1. *Some Previous Applications.* Sequential importance sampling was used in Snijders [69] to sample from the set of all zero-one tables with given row and column sums. The table was built up from left to right, one column at a time; this algorithm could get stuck midway through, forcing it to backtrack or restart.

Chen, Diaconis, Holmes, and Liu [16] introduced the crucial idea of using a combinatorial test to permit only partial fillings which can be completed to full tables. Using this idea, they gave efficient important sampling algorithms for zero-one tables and (2-way) contingency tables. For zero-one tables, the combinatorial test is the Gale-Ryser Theorem. For contingency tables, the test is simpler: the sum of the row sums and the sum of the column sums must agree.

Similarly, our graph algorithm can be viewed as producing *symmetric* zero-one tables with trace 0, where the combinatorial test is the Erdős-Gallai characterization. For both zero-one tables and graphs, refinements to the combinatorial theorems were needed to ensure that partially filled in tables could be completed and sequential probabilities could be computed.

Another interesting sequential importance sampling algorithm is developed in Chen, Dinwoodie and Sullivant [17], to handle multiway contingency tables. A completely different set of mathematical tools turns out to be useful in this case, including Gröbner bases, Markov bases, and toric ideals. Again there is an appealing interplay between combinatorial and algebraic theorems and importance sampling.

These problems fall under a general program: how can one convert a characterization of a combinatorial structure into an efficient generating algorithm? A refinement of the characterization is often needed, for use with testing whether a partially-determined structure can be completed. More algorithms of this nature, including algorithms for generating connected graphs, digraphs, and tournaments, can be found in Blitzstein [12].

8.2. *Importance Sampling of Graphs.* When applying importance sampling with our algorithm, some care is needed because it is possible to generate the same graph in different orders, with different corresponding probabilities. For example, consider the graphical sequence $d = (4, 3, 2, 3, 2)$. The graph with edges

$$\{1, 3\}, \{2, 3\}, \{2, 4\}, \{1, 2\}, \{1, 5\}, \{1, 4\}, \{4, 5\}$$

can be generated by the algorithm in any of 8 orders. For example, there is the order just given, and there is the order

$$\{2, 3\}, \{1, 3\}, \{2, 4\}, \{1, 2\}, \{1, 4\}, \{1, 5\}, \{4, 5\}.$$

The probability of the former is $1/20$ and that of the latter is $3/40$, even though the two sequences correspond to the same graph. This makes it more difficult to directly compute the probability of generating a specific graph, but it does not prevent the importance sampling method from working.

Fix a graphical sequence d of length n as input to the algorithm. We first introduce some notation to clearly distinguish between a graph and a list of edges.

DEFINITION 4. *Let $\mathcal{G}_{n,d}$ be the set of all realizations of d and let $\mathcal{Y}_{n,d}$ be the set of all possible sequences of edges output by the algorithm. For any sequence $Y \in \mathcal{Y}_{n,d}$ of edges, let* $\mathrm{Graph}(Y)$ *be the corresponding graph in $\mathcal{G}_{n,d}$ (with the listed edges and vertex set $\{1, \ldots, n\}$). We call $Y, Y' \in \mathcal{Y}_{n,d}$* equivalent *if* $\mathrm{Graph}(Y') = \mathrm{Graph}(Y)$. *Let $c(Y)$ be the number of $Y' \in \mathcal{Y}_{n,d}$ with* $\mathrm{Graph}(Y') = \mathrm{Graph}(Y)$.

The equivalence relation defined above partitions $\mathcal{Y}_{n,d}$ into equivalence classes. There is an obvious one-to-one correspondence between the equivalence classes and $\mathcal{G}_{n,d}$, and the size of the class of Y is $c(Y)$. Note that $c(Y') = c(Y)$ if Y' is equivalent to Y. The number $c(Y)$ is easy to compute as a product of factorials:

PROPOSITION 2. *Let $Y \in \mathcal{Y}_{n,d}$ and let $i_1, i_2, \ldots, i_m$ be the vertices chosen in the iterations of Step 3 of the algorithm in an instance where Y is output (so the algorithm gives i_1 edges until its degree goes to 0, and then does the same for i_2, etc.). Let $d = d^{(0)}, d^{(1)}, d^{(2)}, \ldots, d^{(j)} = 0$ be the corresponding sequence of graphical degree sequences. Put $i_0 = 0$. Then*

$$c(Y) = \prod_{k=1}^{m} d_{i_k}^{(i_{k-1})}!.$$

PROOF. Let Y' be equivalent to Y. Note from Step 3 of the algorithm that Y' has the same vertex choices $i_1, i_2, \ldots, i_m$ as Y. We can decompose Y and Y' into blocks corresponding to $i_1, \ldots, i_m$. Within each block, Y' and Y must have the same set of edges, possibly in a permuted order. Conversely, Theorem 3 implies that any permutation which independently permutes the edges within each block of the sequence Y yields a sequence Y'' in $\mathcal{Y}_{n,d}$. Clearly any such Y'' is equivalent to Y. □

A main goal in designing our algorithm, in addition to not letting it get stuck, was to have a simple formula for $c(Y)$. In many seemingly similar algorithms, it is difficult to find an analogue of the above formula for $c(Y)$, making it much more difficult to use importance sampling efficiently. Before explaining how importance sampling works in this context, a little more notation is needed.

NOTATION 2. *For $Y \in \mathcal{Y}_{n,d}$, write $\sigma(Y)$ for the probability that the algorithm produces the sequence Y. Given a function f on $\mathcal{G}_{n,d}$, write $\hat{f}$ for the induced function on the larger space $\mathcal{Y}_{n,d}$, with $\hat{f}(Y) = f(\mathrm{Graph}(Y))$.*

Note that for Y the output of a run of the algorithm, $\sigma(Y)$ can easily be computed sequentially along the way. Each time the algorithm chooses an edge, have it record the probability with which it was chosen (conditioned on all of the previously chosen edges), namely, its degree divided by the sum of the degrees of all candidates at that stage. Multiplying these probabilities gives the probability $\sigma(Y)$ of the algorithm producing Y.

We can now show how to do importance sampling with the algorithm, despite having a proposal distribution σ distributed on $\mathcal{Y}_{n,d}$ rather than on $\mathcal{G}_{n,d}$.

PROPOSITION 3. *Let π be a probability distribution on $\mathcal{G}_{n,d}$ and G be a random graph drawn according to π. Let Y be a sequence of edges distributed according to σ. Then*

$$E\left(\frac{\hat{\pi}(Y)}{c(Y)\sigma(Y)}\hat{f}(Y)\right) = Ef(G).$$

In particular, for $Y_1, \ldots, Y_N$ the output sequences of N independent runs of the algorithm,

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N}\frac{\hat{\pi}(Y_i)}{c(Y_i)\sigma(Y_i)}\hat{f}(Y_i)$$

is an unbiased estimator of $Ef(G)$.

PROOF. Let Y be an output of the algorithm. We can compute a sum over $\mathcal{Y}_{n,d}$ by first

summing within each equivalence class and then summing over all equivalence classes:

$$\begin{aligned} E\left(\frac{\hat{\pi}(Y)}{c(Y)\sigma(Y)}\hat{f}(Y)\right) &= \sum_{y\in\mathcal{Y}_{n,d}} \frac{\hat{f}(y)\hat{\pi}(y)}{c(y)\sigma(y)}\sigma(y) \\ &= \sum_{y\in\mathcal{Y}_{n,d}} \frac{\hat{f}(y)\hat{\pi}(y)}{c(y)} \\ &= \sum_{G\in\mathcal{G}_{n,d}} \sum_{y:\mathrm{Graph}(y)=G} \frac{\hat{f}(y)\hat{\pi}(y)}{c(y)} \\ &= \sum_{G\in\mathcal{G}_{n,d}} f(G)\pi(G) \\ &= Ef(G), \end{aligned}$$

where the second to last equality is because on the set $C(G) = \{y : \mathrm{Graph}(y) = G\}$, we have $\hat{f}(y) = f(G), \hat{\pi}(y) = \pi(G)$, and $c(y) = |C(G)|$. □

Taking π to be uniform and f to be a constant function allows us to estimate the number of graphs with degree sequence d; this is explored in the next section. The ratios $W_i = \frac{\hat{\pi}(Y_i)}{c(Y_i)\sigma(Y_i)}$ are called *importance weights*. A crucial feature of our algorithm is that the quantity $c(Y_i)\sigma(Y_i)$ can easily be computed on-the-fly, as described above. By taking f to be a constant function, we see that $EW_i = 1$, so the average sum of the importance weights in $\hat{\mu}$ is N. Another estimator of $Ef(G)$ is then

$$\tilde{\mu} = \frac{1}{\sum_{i=1}^{N} W_i} \sum_{i=1}^{N} W_i \hat{f}(Y_i).$$

This estimator is biased, but often works well in practice and has the advantage that the importance weights need only be known up to a multiplicative constant. Since in practice one often works with distributions that involve unknown normalizing constants, having it suffice to know the importance weights up to a constant is often crucial.

A major factor in the performance of importance sampling is how much variation there is in the importance weights. Let π be the uniform distribution on $\mathcal{G}_{n,d}$. We plot in Figure 4 a histogram of importance weights for 6000 trials with d the degree sequence of the food web of Figure 1. The weights shown are scaled by dividing by 10^{52} and omitting the constant $\hat{\pi}(Y)$. The weights vary greatly from a minimum of 2.9×10^{52} to a maximum of 2.9×10^{58}, but most are between 1.2×10^{56} and 1.9×10^{57}. The ratio of maximum to median is 52, making the largest few weights influential but not completely dominant in importance sampling estimates.

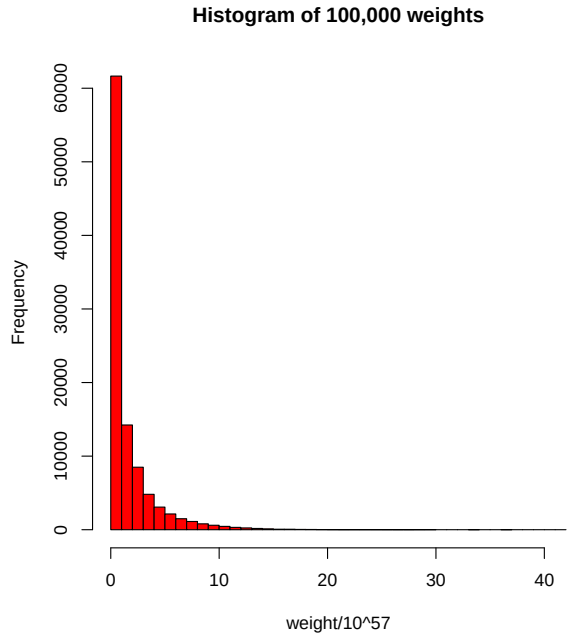

FIG 4. *histogram of 6000 importance weights for the Chesapeake food web*

9. Estimating the Number of Graphs. To estimate the number $|\mathcal{G}_{n,d}|$ of realizations of d, let π be uniform on $\mathcal{G}_{n,d}$ and take f to be the constant function $f(G) = |\mathcal{G}_{n,d}|$. By Proposition 3,

$$E\left(\frac{1}{c(Y)\sigma(Y)}\right) = |\mathcal{G}_{n,d}|.$$

Asymptotic formulas for $|\mathcal{G}_{n,d}|$ are available for regular and some non-regular degree sequences (see Bender and Canfield [9] and McKay and Wormald [52, 54]), but there are few non-asymptotic closed form expressions.

For the food web example of Figure 1, the estimated size of $\mathcal{G}_{n,d}$ was $(1.51 \pm 0.03) \times 10^{57}$ using 6000 trials. The asymptotic formulas are not of much use in such an example, with a fixed n of moderate size (here $n = 33$).

As an application and test of this method, we estimated the number of labeled 3-regular graphs on n vertices for various even values of n. The exact values for all even $n \leq 24$ are available as Sequence A002829 in Sloane's wonderful On-Line Encyclopedia of Integer Sequences [66], and in general they can be computed using a messy recurrence in Goulden and Jackson [32].

For $n = 4$, there is only one labeled 3-regular graph on n vertices, the complete graph K_4. Comfortingly, the algorithm does give 1 as its estimate for this case. In general, a

degree sequence with exactly one realization is called a *threshold sequence* (see [51]), and it is easy to see that the algorithm gives 1 as the estimate for any threshold sequence.

The table below gives the estimators $\hat{\mu}$ obtained by the trials for all even n between 6 and 24, along with the number of trials, the correct value μ, and the percent error. The number after each $\pm$ indicates the estimated standard error.

For each of these degree sequences, the coefficient of variation (ratio of standard deviation to mean) was approximately 0.4, ranging between 0.39 and 0.43. A measure of the efficiency of an importance sampling scheme is the *effective sample size*, as given in [44]. The effective sample size approximates the number of i.i.d. samples from the target distribution required to obtain the same standard error as the importance samples. The estimated effective sample sizes for these examples, computed using the coefficients of variation, range between 422 to 434 for 500 runs of the algorithm.

n	runs	$\hat{\mu}$	μ	% error
6	500	71.1 ± 1.2	70	1.57
8	500	18964 ± 365	19355	2.06
10	500	$(1.126 \pm 0.021) \times 10^7$	1.118×10^7	0.72
12	500	$(1.153 \pm 0.022) \times 10^{10}$	1.156×10^{10}	0.26
14	500	$(1.914 \pm 0.036) \times 10^{13}$	1.951×10^{13}	1.93
16	500	$(5.122 \pm 0.093) \times 10^{16}$	5.026×10^{16}	1.91
18	500	$(1.893 \pm 0.034) \times 10^{20}$	1.877×10^{20}	0.85
20	500	$(9.674 \pm 0.17) \times 10^{23}$	9.763×10^{23}	0.92
22	500	$(6.842 \pm 0.12) \times 10^{27}$	6.840×10^{27}	0.029
24	500	$(6.411 \pm 0.11) \times 10^{31}$	6.287×10^{31}	1.97

As a comparison between choosing candidates uniformly and choosing candidates with probability proportional to their degrees, we generated 50 estimators from each algorithm, with each based on 100 runs of the algorithm applied to the 3-regular degree sequence with $n = 10$. The true value is $11180820 \approx 1.118 \times 10^7$. The mean of the estimators for uniform candidates was 1.137×10^7 (an error of 1.7%), while that of the degree-based selection was 1.121×10^7 (an error of 0.25%).

For a non-regular example, we tested the algorithm with the graphical degree sequence $d = (5, 6, 1, \ldots, 1)$ with eleven 1's. To count the number of labeled graphs with this degree sequence, note that there are $\binom{11}{5} = 462$ such graphs with vertex 1 not joined to vertex 2 by an edge (these graphs look like two separate stars), and there are $\binom{11}{4}\binom{7}{5} = 6930$ such graphs with an edge between vertices 1 and 2 (these look like two joined stars with an isolated edge left over). Thus, the total number of realizations of d is 7392.

Running 500 trials of the algorithm gave the estimate 7176 ± 587, an error of 3%. The algorithm with uniform selection of candidate gave the terrible estimate of 3702 with 500 trials, indicating the importance of choosing a good distribution on the candidate vertices.

10. Trees. With a minor modification, the sequential algorithm can be used to generate random trees with a given degree sequence. The tree algorithm is simpler to analyze and faster than the graph algorithm. Moreover, with an appropriate choice of selection probabilities, the output is exactly uniform. This section uses some standard properties of trees and Prüfer codes, which can be found in many good combinatorics books such as van Lint and Wilson [78].

Throughout this section, let $n \geq 2$ and $d = (d_1, \ldots, d_n)$ be a degree sequence (the case $n = 1$ is trivial). A tree with n vertices has $n - 1$ edges, so it is necessary that $\sum_{i=1}^{n} d_i = 2n - 2$ for a tree with degree sequence d to exist. Also, it is obviously necessary that $d_i \geq 1$ for all i. Conversely, it is easy to check (by induction or using Prüfer codes) that these conditions are sufficient. So for trees, we can use the simple criterion below in place of Erdős-Gallai:

Tree Criterion: There is a tree realizing d if and only if $\sum_{i=1}^{n} d_i = 2n - 2$ and $d_i \geq 1$.

We call a degree sequence satisfying the Tree Criterion *arborical* (as the term "treeical" seemed too reminiscent of molasses).

One simple algorithm uses Prüfer codes. Let d be arborical. Generate a sequence of length $n - 2$ in which i appears $d_i - 1$ times, and take a random permutation of this sequence. The result is the Prüfer code of a uniformly distributed tree with degree sequence d. Of course, using this algorithm requires decoding the Prüfer code to obtain the desired tree.

The appearance of i exactly $d_i - 1$ times in the Prüfer code motivates the following modification of our graph algorithm. In the tree algorithm below, we forbid creating an edge between two vertices of degree 1 (except for the final edge), and choose vertices with probability proportional to the degree minus 1.

Sequential Algorithm For Random Tree with Given Degrees

Input: an arborical sequence $(d_1, ..., d_n)$.

1. Let E be an empty list of edges.
2. If d is 0 except at $i \neq j$ with $d_i = d_j = 1$, add $\{i, j\}$ to E and terminate.
3. Choose the least i with $d_i = 1$.
4. Pick j of degree at least 2, with probability proportional to $d_j - 1$.
5. Add the edge $\{i, j\}$ to E and update d to $\ominus_{i,j} d$.
6. Return to step 2.

Output: E.

We need to show that the restriction against connecting degree 1 to degree 1 does not allow the tree algorithm to get stuck, and that the output is always a tree.

Theorem 4. *Given an arborical sequence $d = (d_1, \ldots, d_n)$ as input, the algorithm above terminates with a tree realizing d.*

PROOF. Note that there is no danger of creating multiple edges since except for the final edge, each new edge joins a vertex of degree 1 to a vertex of degree greater than 1, after which the degree 1 vertex is updated to degree 0 and not used again. We induct on n. For $n = 2$, the only possible degree sequence is $(1, 1)$, and the algorithm immediately terminates with the tree whose only edge is $\{1, 2\}$. Assume that $n \geq 3$ and the claim holds for $n - 1$.

The sequence d must contain at least two 1's, since otherwise the degree sum would be at least $2n - 1$ (trees have leaves!). And d must contain at least one entry greater than 1, since otherwise the degree sum would be $n < 2n - 2$. Thus, the algorithm can complete the first iteration, choosing an edge $\{i_0, j_0\}$ where $d_{i_0} = 1, d_{j_0} > 1$. The algorithm then proceeds to operate on $\ominus_{i_0,j_0} d$.

Let d' be obtained from $\ominus_{i_0,j_0} d$ by deleting coordinate i_0, which is the position of the only 0 in $\ominus_{i_0,j_0} d$. Index the coordinates in d' as they were indexed in d. Then $\sum_{i=1}^{n-1} d'_i = 2(n-1) - 2$ and $d'_i \geq 1$, so the inductive hypothesis applies to d'. Running the algorithm with d' as input gives a tree T' realizing d', with $\{i_0, j_0\}$ not an edge since T' does not even have a vertex labeled i_0. Creating a leaf labeled i_0 in T' with an edge $\{i_0, j_0\}$ yields a tree realizing d. Thus, the algorithm produces a tree of the desired form. □

It is well-known and easy to check (again using induction or Prüfer codes; see Equation (2.4) in [78]) that for any d, the number of labeled trees realizing d is the multinomial coefficient

$$N_{\text{trees}}(d_1, \ldots, d_n) = \binom{n-2}{d_1 - 1, \ldots, d_n - 1}.$$

We now show that the tree algorithm produces an exactly uniform tree.

PROPOSITION 4. *Let d be an arborical sequence. The output of the tree algorithm is a tree which is uniformly distributed among the $N_{trees}(d)$ trees realizing d.*

PROOF. Let T be the random tree produced by the algorithm. There is only one order of edge generation which can produce T because at each stage a vertex of degree 1 is used. So it suffices to find the probability of the algorithm generating the specific order of edges corresponding to T. Each vertex i with $d_i > 1$ is chosen repeatedly (not necessarily consecutively) until its degree is reduced to 1, each time with probability proportional to $d_i - 1$. The normalizing constant at each stage (except for the final joining of two 1's) is $\sum_i (d_i - 1)$, summed over all i with $d_i > 1$. This quantity is initially $n - 2$ and decreases by 1 after each edge is picked, until it reaches a value of 1. Thus, the probability of T being generated is

$$\frac{(d_1 - 1)!(d_2 - 1)! \cdots (d_n - 1)!}{(n-2)!} = 1 \Big/ \binom{n-2}{d_1 - 1, \ldots, d_n - 1},$$

which shows that T is uniformly distributed. □

11. An Exponential Model. For a labeled graph G with n vertices, let $d_i(G)$ be the degree of vertex i. In the preceding sections, we have kept d_i fixed. In this section, we allow d_i to be random by putting an exponential model on the space of all labeled graphs with n vertices. In this model, the d_i are used as energies or sufficient statistics.

More formally, define a probability measure P_β on the space of all graphs on n vertices by

$$P_\beta(G) = Z^{-1} \exp\left(-\sum_{i=1}^{n} \beta_i d_i(G) \right),$$

where Z is a normalizing constant. The real parameters $\beta_1, \ldots, \beta_n$ are chosen to achieve given expected degrees. This model appears explicitly in Park and Newman [61], using the tools and language of statistical mechanics.

Holland and Leinhardt [37] give iterative algorithms for the maximum likelihood estimators of the parameters, and Snijders [67] considers MCMC methods. Preliminary work with Persi Diaconis and Sourav Chatterjee seems to give an efficient and accurate method of computing the MLE of the β_i for both directed and undirected cases. Techniques of Haberman [33] can be used to prove that the maximum likelihood estimates of the β_i are consistent and asymptotically normal as $n \to \infty$, provided that there is a constant B such that $|\beta_i| \le B$ for all i.

Such exponential models are standard fare in statistics, statistical mechanics, and social networking (where they are called p^* models). They are used for directed graphs in Holland and Leinhardt [37] and for graphs in Frank and Strauss [29, 72] and Snijders [67, 68], with a variety of sufficient statistics (see the surveys in [3], [58], and [68]). One standard motivation for using the probability measure P_β when the degree sequence is the main feature of interest is that this model gives the maximum entropy distribution on graphs with a given expected degree sequence (see Lauritzen [45] for further discussion of this). Unlike most other exponential models on graphs, the normalizing constant Z is available in closed form. Furthermore, there is an easy method of sampling exactly from P_β, as shown by the following. The same formulas are given in [61], but for completeness we provide a brief proof.

LEMMA 1. *Fix real parameters $\beta_1, \ldots, \beta_n$. Let Y_{ij} be independent binary random variables for $1 \le i < j \le n$, with*

$$P(Y_{ij} = 1) = \frac{e^{-(\beta_i + \beta_j)}}{1 + e^{-(\beta_i + \beta_j)}} = 1 - P(Y_{ij} = 0).$$

Form a random graph G by creating an edge between i and j if and only if $Y_{ij} = 1$. Then G is distributed according to P_β, with

$$Z = \prod_{1 \le i < j \le n} (1 + e^{-(\beta_i + \beta_j)}).$$

PROOF. Let G be a graph and $y_{ij} = 1$ if $\{i, j\}$ is an edge of G, $y_{ij} = 0$ otherwise. Then the probability of G under the above procedure is

$$P(Y_{ij} = y_{ij} \text{ for all } i, j) \quad = \quad \prod_{i<j} \frac{e^{-y_{ij}(\beta_i+\beta_j)}}{1 + e^{-(\beta_i+\beta_j)}}.$$

The denominator in this expression is the claimed Z. The numerator simplifies to $e^{-\sum_{i=1}^{n} \beta_i d_i(G)}$ since we can restrict the product to factors with $y_{ij} = 1$, and then each edge $\{i, j\}$ contributes 1 to the coefficients of both i and j. □

Note that putting $\beta_i = 0$ results in the uniform distribution on graphs. Also, it follows from Lemma 1 above that choosing $\beta_i = \beta$ for all i, for $\beta = -\frac{1}{2}\log(\frac{p}{1-p})$, we recover the classical Erdős-Rényi model.

Another model with given expected degrees and edges chosen independently is considered in Chung and Lu [19, 20], where an edge between i and j is created with probability $w_i w_j / \sum_k w_k$ (including the case $i = j$), where w_i is the desired expected degree of vertex i and it is assumed that $w_i^2 < \sum_k w_k$ for all i. This has the advantage that it is immediate that the expected degree of vertex i is w_i, without any parameter estimation required. The exponential model of this section makes it more difficult to choose the β_i, but it yields the maximum entropy distribution, makes the degrees sufficient statistics, and does not require the use of loops. If loops are desired in the exponential model, they may easily be added by allowing $i = j$ in Lemma 1. We would like to better understand the precise relationship between the distribution obtained from the Chung and Lu model and the maximum entropy distribution of the exponential model.

We remark that the formula for the normalizing constant is equivalent to the identity

$$\prod_{1\le i<j\le n} (1 + x_i x_j) = \sum_G \prod_{i=1}^{n} x_i^{d_i(G)},$$

where the sum on the right is over all graphs G on vertices $1, \ldots, n$. This identity is closely related to the following symmetric function identity (which is a consequence of Weyl's identity for root systems; see Exercise 9 in Section I.5 of Macdonald [50]):

$$\prod_{1\le i<j\le n} (1 + x_i x_j) = \sum_\lambda s_\lambda,$$

where s_λ is the Schur function corresponding to a partition λ, and the sum on the right ranges over all partitions λ with Frobenius notation of the form $(\alpha_1 - 1 \cdots \alpha_r - 1 | \alpha_1 \cdots \alpha_r)$, with $\alpha_1 \le n - 1$. We hope to find (and use) a stochastic interpretation for this Schur function expansion.

The exponential model given by P_β is quite rich; it has n real parameters. We can test the suitability of the model (with unspecified parameters) by conditioning on the degrees

and using the following lemma (which follows easily from the fact that $P_\beta(G)$ depends on G only through its degrees).

LEMMA 2. *If a graph G is chosen according to P_β for any fixed choice of the parameters $\beta_1, \ldots, \beta_n$, the conditional distribution of G given its degrees $d_1(G), \ldots, d_n(G)$ is uniform over all graphs with this degree sequence.*

Thus, we may test the exponential model P_β given a specific graph G with degrees $d_i(G)$ as follows. First choose any test statistic T (such as the number of triangles, the diameter, the number of components, or the size of the largest component), and compute $T(G)$. Then generate a large number of uniform graphs with the same degrees as G, and compare the observed value of $T(G)$ against the distribution of T obtained from the sampled random graphs. For history and background on these conditional tests, see Diaconis and Sturmfels [24]. An alternative approach to testing such a model is to embed it in a larger family, as in Holland and Leinhardt [37]. Indeed, it is shown in Section 4.4 of Lehmann and Romano [46] that our proposed test is optimal (UMP unbiased) in the exponential model extended by adding $T(G)$ to the sufficient statistic $(d_1(G), \ldots, d_n(G))$.

As an example, we return now to the Chesapeake Bay food web shown in Figure 1. The graph has 33 vertices including at least one of every degree from 1 to 10, as illustrated below. Note that the degrees vary widely but do not resemble a power-law distribution.

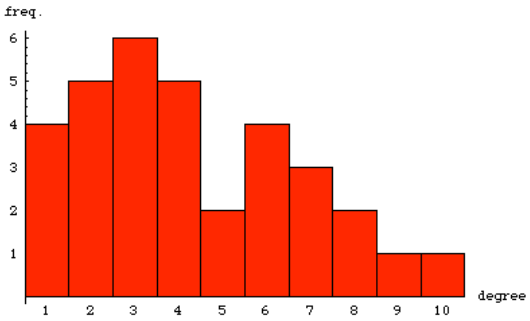

FIG 5. *frequencies of degrees in Chesapeake food web*

As a first test statistic, we computed the *clustering coefficient* of each graph. Intuitively, the clustering coefficient measures how cliquish a network is, in the sense of how likely it is for two neighbors of the same vertex to be connected. There are a few different definitions in use, but we will use the original definition of Watts and Strogatz [80]: for each vertex v of degree $d_v \geq 2$, let C_v be the proportion of edges present between neighbors of v out of the $\binom{d_v}{2}$ possible edges. Put $C_v = 0$ if $d_v < 2$. The clustering coefficient of a graph is then defined to be the average of C_v over all vertices of the graph.

Using the estimator $\tilde{\mu}$ of Section 8.2, the estimated average clustering coefficient for a graph with the same degrees as the food web is 0.157. A histogram of the estimated distribution of the clustering coefficient is shown below. To generate each bin, we again used the estimator $\tilde{\mu}$, with f the indicator function of the clustering coefficient falling into that interval.

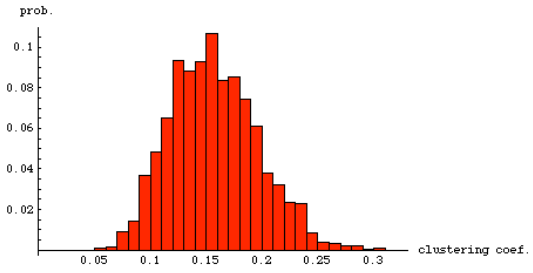

FIG 6. *histogram of clustering coefficients; real food web value is* 0.176

The actual clustering coefficient of the food web is 0.176, slightly above the mean. Thus, the clustering coefficient agrees well with the predictions of the exponential model.

In an attempt to explain and quantify the observation that the actual food web appeared more compact and neatly ordered than most of the random graphs such as those shown in Figures 2 and 3, we decided to look at cycles in the graphs. Specifically, we counted the number of k-cycles in the real food web and the first 1000 random graphs, for $3 \le k \le 6$. The cycles are treated as unoriented subgraphs of the graph, i.e., a cycle $x \to y \to z \to x$ is considered the same cycle as $y \to z \to x \to y$ and $x \to z \to y \to x$.

The enumeration was done by recursively counting the number of simple paths from v to v, summing over all v, and dividing by $2k$ since each cycle can be started at any of its k vertices and traversed in either direction. Histograms of the numbers of k-cycles are shown in Figure 7.

For 3-cycles (triangles), the actual value of 18 is extremely close to the estimated mean, which is 19. This is not surprising, especially since the clustering coefficient is closely related to the number of triangles.

For 4-cycles though, the actual value of 119 is nearly double the estimated mean of 60. In fact, the number of 4-cycles in the actual food web is larger than the number of 4-cycles in *any* of the 1000 random graphs tested! Explaining this of course requires looking at the directed graph, but the undirected graph was very helpful in detecting this phenomenon in the first place. Inspecting the corresponding directed subgraphs, two forms are prevalent: (1) x and y both eat z and w and (2) x eats y and z, while y and z eat w. Interestingly, Milo et al. [55] observe that pattern (2) is extremely common in all seven of the food webs they study. They call a pattern which occurs much more often in real networks of some

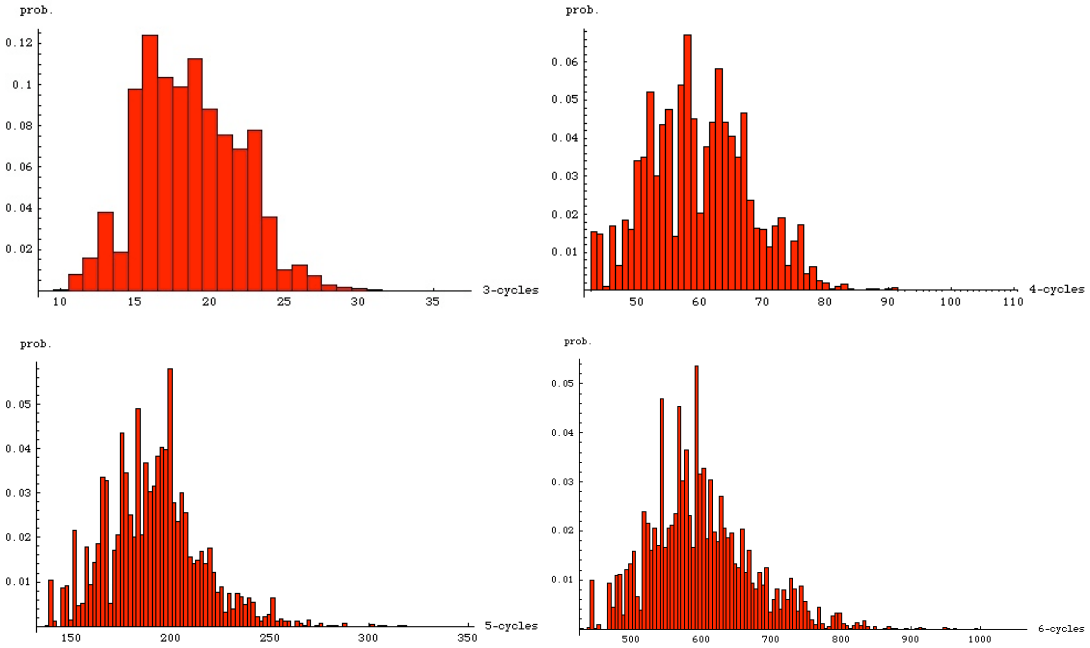

FIG 7. *histograms of k-cycle counts; real food web has 18, 119, 153, 582 respectively*

kind than in the corresponding random networks a "network motif" (see also Itzkovitz et al. [38] for more on network motifs). Finding network motifs can reveal structure in the network which would be missed by taking a highly degree-centric point of view.

In generating random graphs for comparison purposes, Milo et al. use two algorithms. First, they use the switchings Markov chain discussed in Section 3.5 (adapted for directed graphs), which is not known to be rapidly mixing for general degree sequences. Second, they use a variant of the pairing model, modified from an algorithm in Newman, Strogatz and Watts [59]. Their algorithm sometimes gets stuck and the output is non-uniform, as explained in King [43]. Our algorithm also has a non-uniform output distribution, but never gets stuck and makes it easy to estimate with respect to the uniform distribution, provided that undirected graphs can be used (which depends on the specific application).

Returning to the cycle results, for 5-cycles the actual value is 153, which is significantly lower than the estimated mean of 191. It is at the 5th percentile of the estimated distribution.

For 6-cycles, the actual value of 582 is close to the estimated mean of 595. This was rather surprising, as the intuition that the food web is fairly hierarchical (the big fish eats the small fish) would suggest that there would be few long or moderately long cycles in the real graph. A biological interpretation would be welcome for why 4-cycles are extremely common and 5-cycles are rare, while 6-cycles are close to the middle of the distribution.

Software. Graphs were drawn using Graphviz [26]. The implementation of the algorithm and importance sampling computations presented here were done using Mathematica [81] and R [62] on Mac OS X. Source code for the algorithm is freely available by making an e-mail request to the first author.

Acknowledgements. We thank Alex Gamburd, Susan Holmes, Brendan McKay, Richard Stanley, and Nick Wormald for their helpful suggestions and references.

REFERENCES

[1] Aiello, W., Chung, F., and Lu, L. (2001). A random graph model for power law graphs. *Experiment. Math.* **10**, 1, 53–66.

[2] Aiello, W., Chung, F., and Lu, L. (2002). Random evolution in massive graphs. In *Handbook of massive data sets*. Massive Comput., Vol. **4**. Kluwer Acad. Publ., Dordrecht, 97–122.

[3] Anderson, C., Wasserman, S., and Crouch, B. (1999). A p^* primer: Logit models for social networks. *Social Networks 21*, 37–66.

[4] Arratia, R. and Liggett, T. M. (2005). How likely is an i.i.d. degree sequence to be graphical? *Ann. Appl. Probab.* **15**, 1B, 652–670.

[5] Baird, D. and Ulanowicz, R. E. (1989). The seasonal dynamics of the Chesapeake bay ecosystem. *Ecological Monographs 59*, 329–364.

[6] Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science 286*, 509–512.

[7] Bassetti, F. and Diaconis, P. (2005). Examples comparing importance sampling and the Metropolis algorithm. To appear, Illinois Journal of Mathematics.

[8] Bayati, M. and Saberi, A. (2006). Fast generation of random graphs via sequential importance sampling. Preprint.

[9] Bender, E. A. and Canfield, E. R. (1978). The asymptotic number of labeled graphs with given degree sequences. *J. Combinatorial Theory Ser. A* **24**, 3, 296–307.

[10] Berge, C. (1976). *Graphs and Hypergraphs*. Elsevier, New York.

[11] Bezáková, I., Sinclair, A., Štefankovič, D., and Vigoda, E. (2006). Analysis of sequential importance sampling for contingency tables. Preprint.

[12] Blitzstein, J. K. (2006). Building random objects sequentially: From characterization to algorithm. Ph.D. thesis, Stanford University. In preparation.

[13] Bollobás, B. (1980). A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European J. Combin.* **1**, 4, 311–316.

[14] Bollobás, B., Riordan, O., Spencer, J., and Tusnády, G. (2001). The degree sequence of a scale-free random graph process. *Random Structures Algorithms* **18**, 3, 279–290.

[15] Britton, T., Deijfen, M., and Martin-Löf, A. (2005). Generating simple random graphs with prescribed degree distribution. Preprint.

[16] Chen, Y., Diaconis, P., Holmes, S., and Liu, J. S. (2005). Sequential Monte Carlo methods for statistical analysis of tables. *Journal of the American Statistical Association 100*, 109–120.

[17] Chen, Y., Dinwoodie, I., and Sullivant, S. (2005). Sequential importance sampling for multiway tables. *Annals of Statistics (to appear)*.

[18] Choudum, S. A. (1986). A simple proof of the Erdős-Gallai theorem on graph sequences. *Bull. Austral. Math. Soc.* **33**, 1, 67–70.

[19] Chung, F. and Lu, L. (2002a). The average distances in random graphs with given expected degrees. *Proc. Natl. Acad. Sci. USA* **99**, 25, 15879–15882 (electronic).

[20] Chung, F. and Lu, L. (2002b). Connected components in random graphs with given expected degree sequences. *Ann. Comb.* **6**, 2, 125–145.

[21] Cooper, C., Dyer, M., and Greenhill, C. (2005). Sampling regular graphs and a peer-to-peer network. To appear, Combinatorics, Probability and Computing.
[22] Cooper, C. and Frieze, A. (2003). A general model of web graphs. *Random Structures Algorithms* **22**, 3, 311–335.
[23] Diaconis, P. and Gangolli, A. (1995). Rectangular arrays with fixed margins. In *Discrete probability and algorithms (Minneapolis, MN, 1993)*. IMA Vol. Math. Appl., Vol. **72**. Springer, New York, 15–41.
[24] Diaconis, P. and Sturmfels, B. (1998). Algebraic algorithms for sampling from conditional distributions. *Ann. Statist.* **26**, 1, 363–397.
[25] Doucet, A., de Freitas, N., and Gordon, N. (2001). An introduction to sequential Monte Carlo methods. In *Sequential Monte Carlo methods in practice*. Stat. Eng. Inf. Sci. Springer, New York, 3–14.
[26] Ellson, J., Gansner, E., Koutsofios, E., North, S., and Woodhull, G. (2003). Graphviz and dynagraph – static and dynamic graph drawing tools. In *Graph Drawing Software*, M. Junger and P. Mutzel, Eds. Springer-Verlag, Berlin, 127–148.
[27] Erdős, P. and Gallai, T. (1960). Graphen mit punkten vorgeschriebenen grades. *Mat. Lapok 11*, 264–274.
[28] Fishman, G. S. (1996). *Monte Carlo: Concepts, Algorithms, and Applications*. Springer Series in Operations Research. Springer-Verlag, New York.
[29] Frank, O. and Strauss, D. (1986). Markov graphs. *J. Amer. Statist. Assoc.* **81**, 395, 832–842.
[30] Gamburd, A. (2005). Poisson-Dirichlet distribution for random Belyi surfaces. Preprint.
[31] Gkantsidis, C., Mihail, M., and Zegura, E. (2003). The Markov chain simulation method for generating connected power law random graphs. In *Proc. 5th Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM, Philadelphia.
[32] Goulden, I. P. and Jackson, D. M. (2004). *Combinatorial enumeration*. Dover Publications Inc., Mineola, NY.
[33] Haberman, S. J. (1977). Maximum likelihood estimates in exponential response models. *Ann. Statist.* **5**, 5, 815–841.
[34] Hakimi, S. L. (1962). On realizability of a set of integers as degrees of the vertices of a linear graph. I. *J. Soc. Indust. Appl. Math. 10*, 496–506.
[35] Hammersley, J. M. and Handscomb, D. C. (1965). *Monte Carlo methods*. Methuen & Co. Ltd., London.
[36] Havel, V. (1955). A remark on the existence of finite graphs. *Časopis Pěst. Mat. 80*, 477–480.
[37] Holland, P. W. and Leinhardt, S. (1981). An exponential family of probability distributions for directed graphs. *J. Amer. Statist. Assoc.* **76**, 373, 33–65.
[38] Itzkovitz, S., Milo, R., Kashtan, N., Ziv, G., and Alon, U. (2003). Subgraphs in random networks. *Phys. Rev. E (3)* **68**, 2, 026127, 8.
[39] Jerrum, M. and Sinclair, A. (1990). Fast uniform generation of regular graphs. *Theoret. Comput. Sci.* **73**, 1, 91–100.
[40] Jerrum, M., Sinclair, A., and McKay, B. (1992). When is a graphical sequence stable? In *Random graphs, Vol. 2 (Poznań, 1989)*. Wiley-Intersci. Publ. Wiley, New York, 101–115.
[41] Kim, J. H. and Vu, V. H. (2003). Generating random regular graphs. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*. ACM, New York, 213–222 (electronic).
[42] Kim, J. H. and Vu, V. H. (2004). Sandwiching random graphs: universality between random graph models. *Adv. Math.* **188**, 2, 444–469.
[43] King, O. D. (2004). Comment on: "Subgraphs in random networks". *Phys. Rev. E* **70**, 5, 058101, 3.
[44] Kong, A., Liu, J. S., and Wong, W. H. (1994). Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association* **89**, 425, 278–288.
[45] Lauritzen, S. L. (1988). *Extremal families and systems of sufficient statistics*. Lecture Notes in Statistics, Vol. **49**. Springer-Verlag, New York.
[46] Lehmann, E. L. and Romano, J. P. (2005). *Testing statistical hypotheses*. Springer Texts in Statistics. Springer, New York.

[47] Liu, J. S. (2001). *Monte Carlo strategies in scientific computing*. Springer Series in Statistics. Springer-Verlag, New York.
[48] Liu, J. S. and Chen, R. (1998). Sequential Monte Carlo methods for dynamic systems. *J. Amer. Statist. Assoc.* **93**, 443, 1032–1044.
[49] Lovász, L. and Plummer, M. D. (1986). *Matching theory*. North-Holland Mathematics Studies, Vol. **121**. North-Holland Publishing Co., Amsterdam.
[50] Macdonald, I. G. (1995). *Symmetric functions and Hall polynomials*. Oxford Mathematical Monographs. The Clarendon Press Oxford University Press, New York.
[51] Mahadev, N. V. R. and Peled, U. N. (1995). *Threshold graphs and related topics*. Annals of Discrete Mathematics, Vol. **56**. North-Holland Publishing Co., Amsterdam.
[52] McKay, B. D. and Wormald, N. C. (1990a). Asymptotic enumeration by degree sequence of graphs of high degree. *European J. Combin.* **11**, 6, 565–580.
[53] McKay, B. D. and Wormald, N. C. (1990b). Uniform generation of random regular graphs of moderate degree. *J. Algorithms* **11**, 1, 52–67.
[54] McKay, B. D. and Wormald, N. C. (1991). Asymptotic enumeration by degree sequence of graphs with degrees $o(n^{1/2})$. *Combinatorica* **11**, 4, 369–382.
[55] Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. (2002). Network motifs: simple building blocks of complex networks. *Science 298*, 824–827.
[56] Molloy, M. and Reed, B. (1995). A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms* **6**, 2-3, 161–179.
[57] Molloy, M. and Reed, B. (1998). The size of the giant component of a random graph with a given degree sequence. *Combin. Probab. Comput.* **7**, 3, 295–305.
[58] Newman, M. (2003). The structure and function of complex networks. *SIAM Review 45*, 167–256.
[59] Newman, M. E. J., Strogatz, S., and Watts, D. (2001). Random graphs with arbitrary degree distributions and their applications. *Physical Review E 64*, 026118.
[60] Owen, A. and Zhou, Y. (2000). Safe and effective importance sampling. *Journal of the American Statistical Association* **95**, 449, 135–143.
[61] Park, J. and Newman, M. E. J. (2004). Statistical mechanics of networks. *Phys. Rev. E (3)* **70**, 6, 066117, 13.
[62] R Development Core Team. (2005). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna.
[63] Robalewska, H. D. and Wormald, N. C. (2000). Random star processes. *Combin. Probab. Comput.* **9**, 1, 33–43.
[64] Rubinstein, R. Y. and Kroese, D. P. (2004). *The cross-entropy method*. Information Science and Statistics. Springer-Verlag, New York.
[65] Ruciński, A. and Wormald, N. C. (1992). Random graph processes with degree restrictions. *Combin. Probab. Comput.* **1**, 2, 169–180.
[66] Sloane, N. J. A. (2005). The on-line encyclopedia of integer sequences, published electronically at http://www.research.att.com/∼njas/sequences/.
[67] Snijders, T. (2002). Markov chain Monte Carlo estimation of exponential random graph models. *Journal of Social Structure* **3**, 2.
[68] Snijders, T., Pattison, P., Robins, G., and Handcock, M. (2004). New specifications for exponential random graph models. To appear, Sociological Methodology.
[69] Snijders, T. A. B. (1991). Enumeration and simulation methods for 0-1 matrices with given marginals. *Psychometrika* **56**, 3, 397–417.
[70] Stanley, R. P. (1999). *Enumerative combinatorics. Vol. 2*. Cambridge University Press, Cambridge.
[71] Steger, A. and Wormald, N. C. (1999). Generating random regular graphs quickly. *Combin. Probab. Comput.* **8**, 4, 377–396.
[72] Strauss, D. (1986). On a general class of models for interaction. *SIAM Review* **28**, 4, 513–527.
[73] Taylor, R. (1982). Switchings constrained to 2-connectivity in simple graphs. *SIAM J. Algebraic*

Discrete Methods **3**, 1, 114–121.
[74] Tinhofer, G. (1979). On the generation of random graphs with given properties and known distribution. *Applied Computer Science Berichte Praktische Informatik 13*, 265–296.
[75] Tinhofer, G. (1990). Generating graphs uniformly at random. *Computing Supplementum 7*, 235–255.
[76] Tripathi, A. and Vijay, S. (2003). A note on a theorem of Erdős & Gallai. *Discrete Math.* **265**, 1-3, 417–420.
[77] Ulanowicz, R. E. (2005). Ecosystem network analysis web page, published electronically at http://www.cbl.umces.edu/∼ulan/ntwk/network.html.
[78] van Lint, J. H. and Wilson, R. M. (2001). *A course in combinatorics*, Second ed. Cambridge University Press, Cambridge.
[79] Viger, F. and Latapy, M. (2005). Fast generation of random connected graphs with prescribed degrees. Preprint.
[80] Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature* **393**, 6684, 440–442.
[81] Wolfram Research. (2001). Mathematica, Version 4.1.
[82] Wormald, N. C. (1984). Generating random regular graphs. *J. Algorithms* **5**, 2, 247–280.
[83] Wormald, N. C. (1999). Models of random regular graphs. In *Surveys in combinatorics, 1999 (Canterbury)*. London Math. Soc. Lecture Note Ser., Vol. **267**. Cambridge Univ. Press, Cambridge, 239–298.

Department of Mathematics
Stanford, CA 94305
e-mail: jkb@math.stanford.edu

Departments of Mathematics and Statistics
Stanford, CA 94305